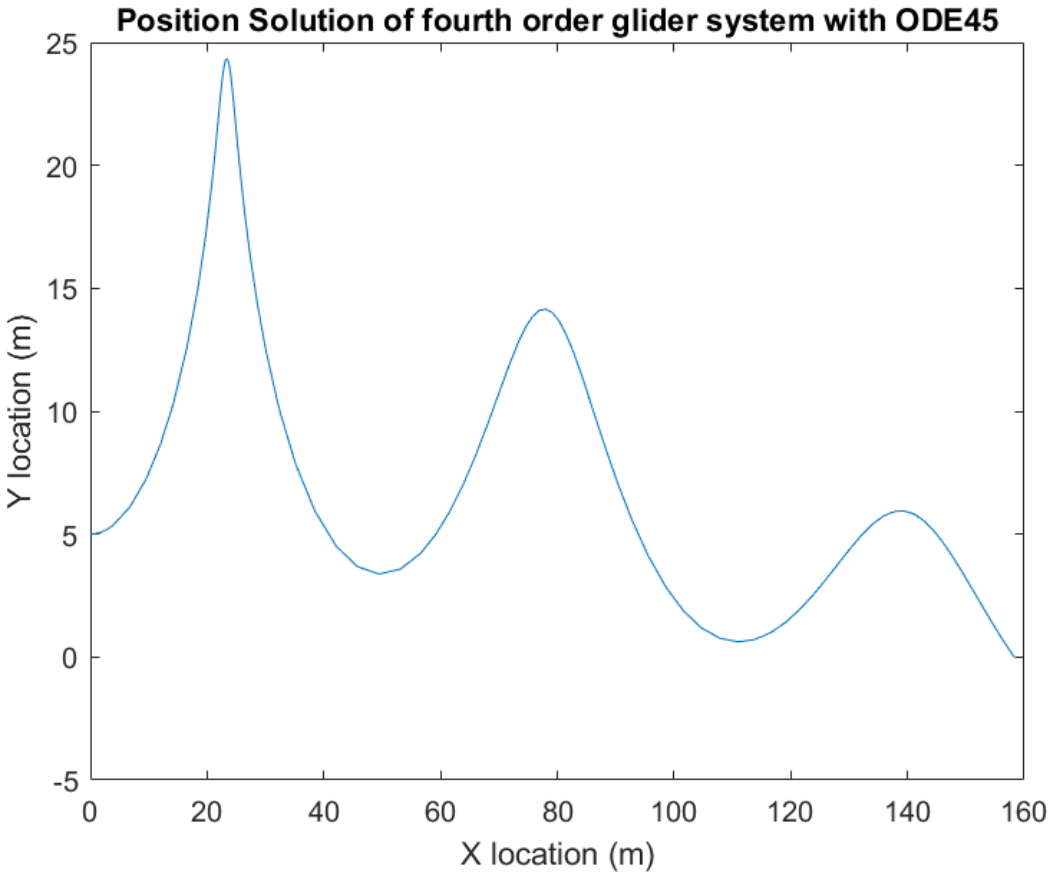
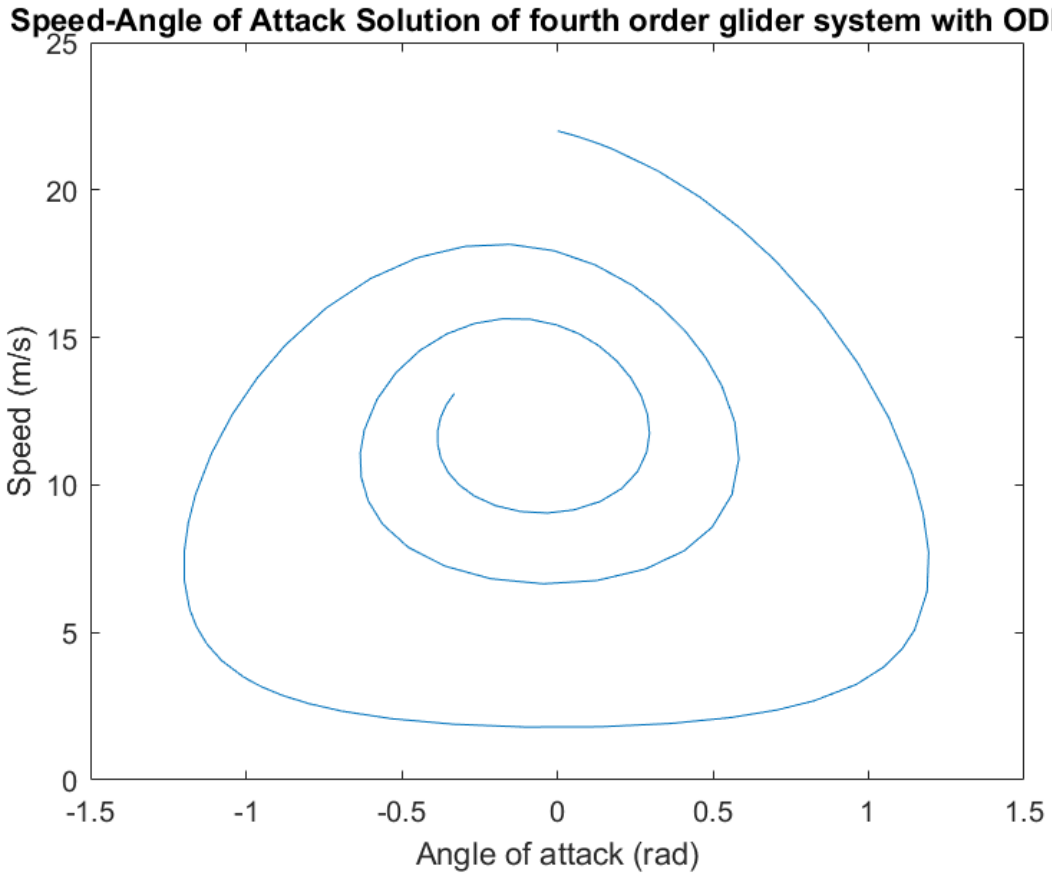
16.90 Report Project 1

By Codrin Oneci

1.

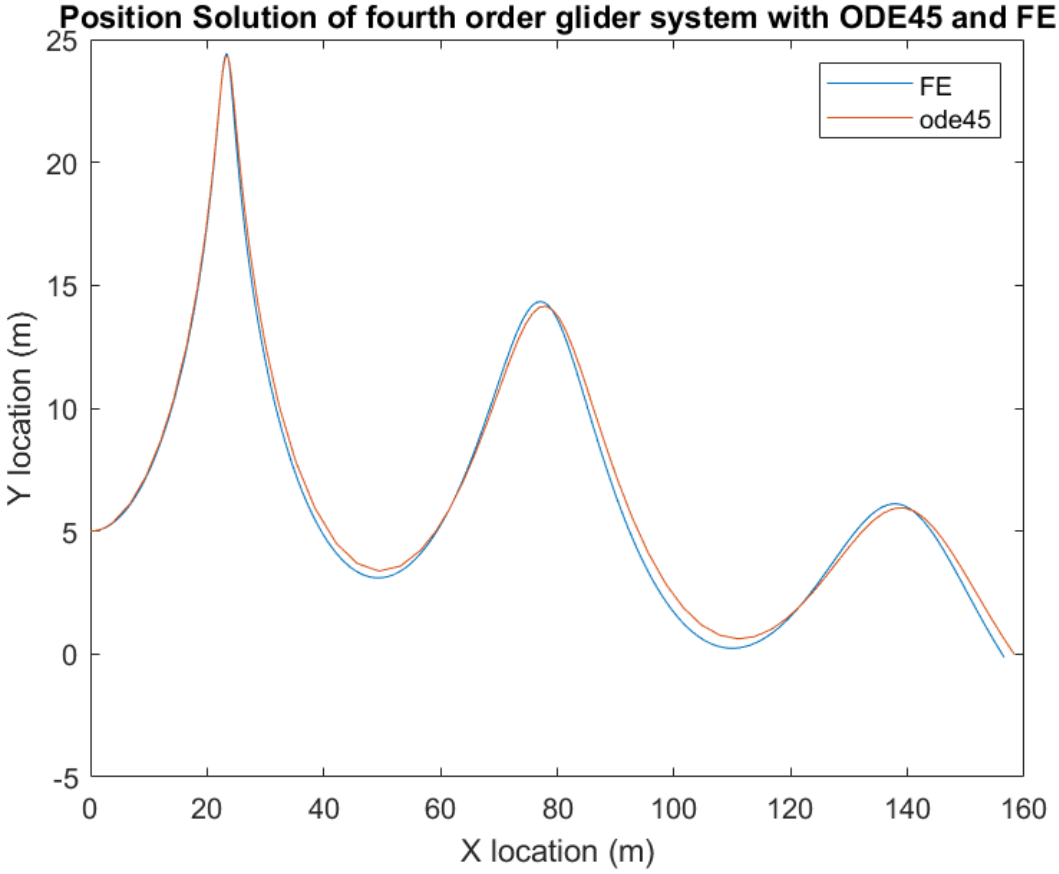


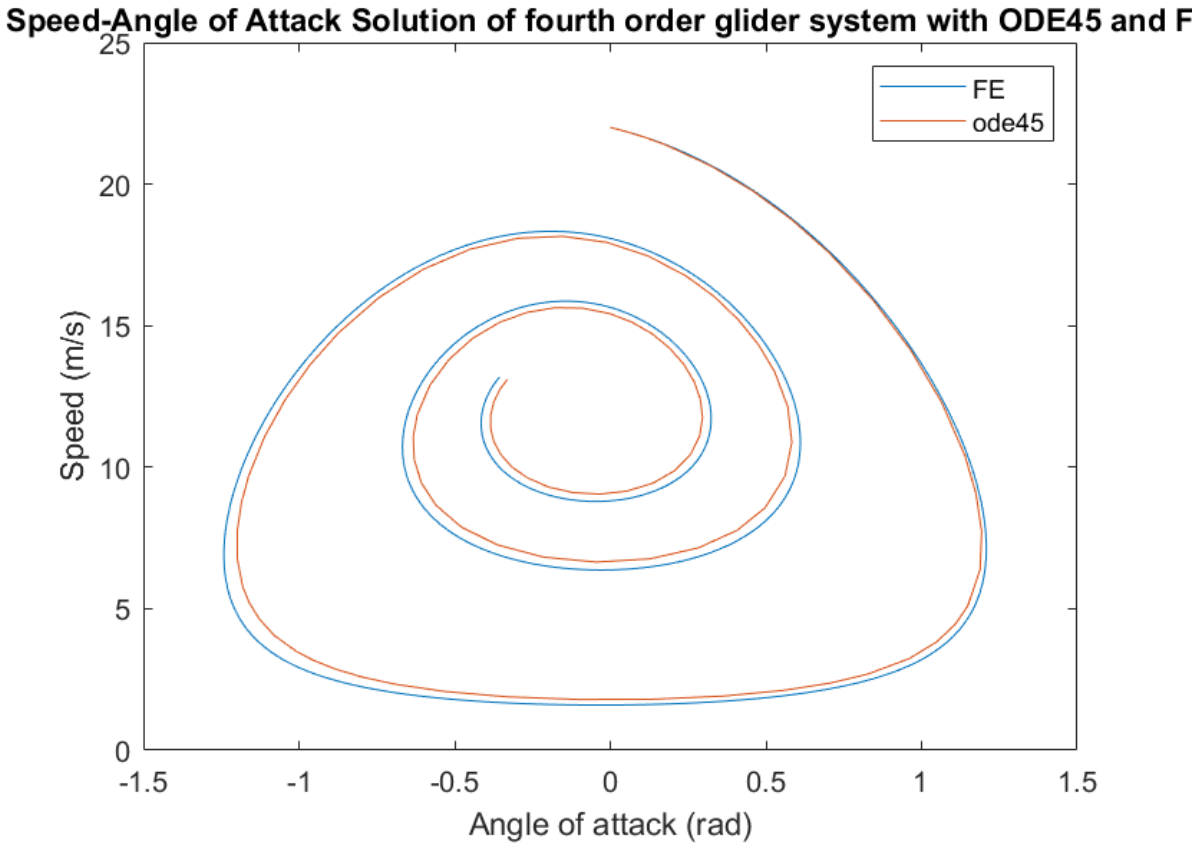


Observe that in both plots the amplitudes of oscillations of the parameters (3 of the degrees of freedom) are decreasing. For this initialization with state vector u0= <22, 0, 0, 5> the x-coordinate is monotonically increasing. We would expect always to have decaying oscillations: physically this would be interpreted as an effect of friction. I will show in this report that the eigenvalues of the linearized ODE problem have negative real parts. As expected the periods in which the angle of attack is small can be described by small changes of the instantaneous speed of the glider. In an oscillation, a high speed of the glider causes an abrupt change in the y-coordinate (and angle of attack) due to the nonlinearity of the lift function in its dependence on air-speed. Both plotted curves are not closed and the beginning and end states are evident.

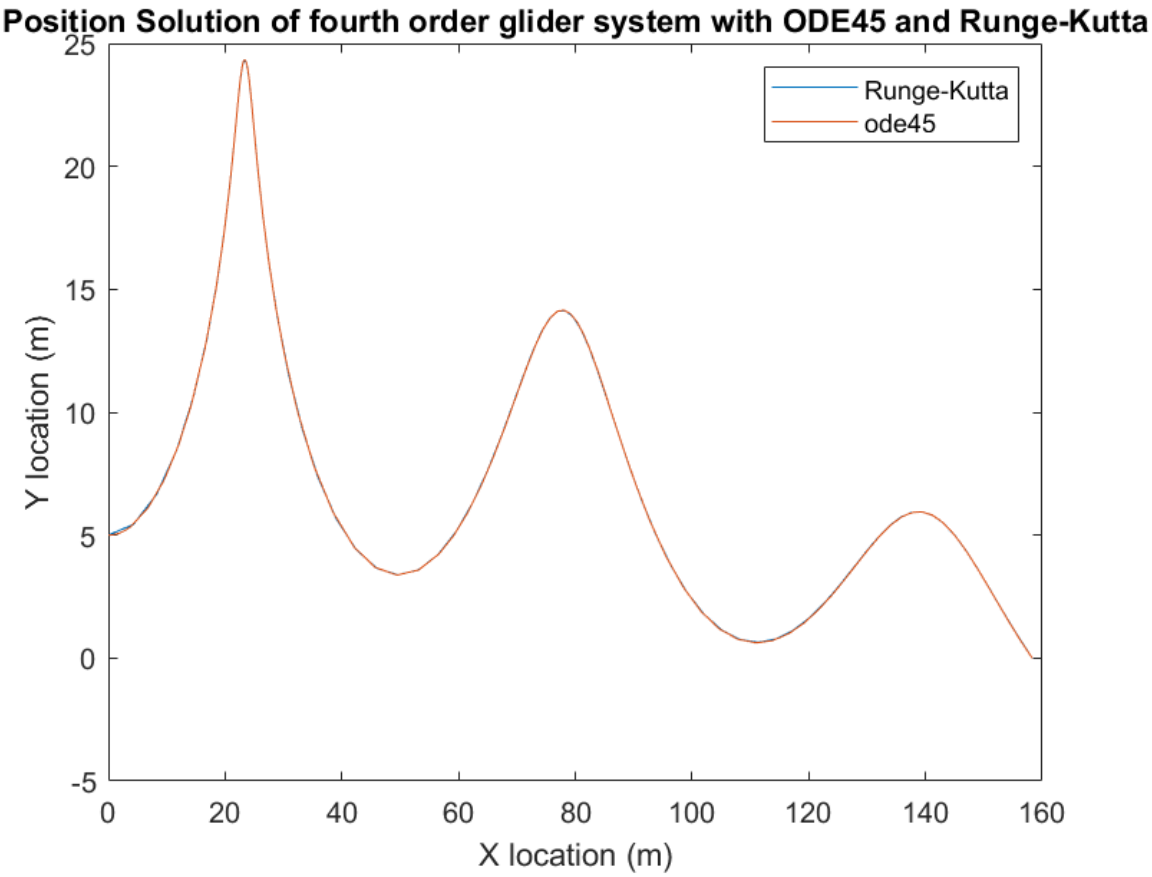
2.

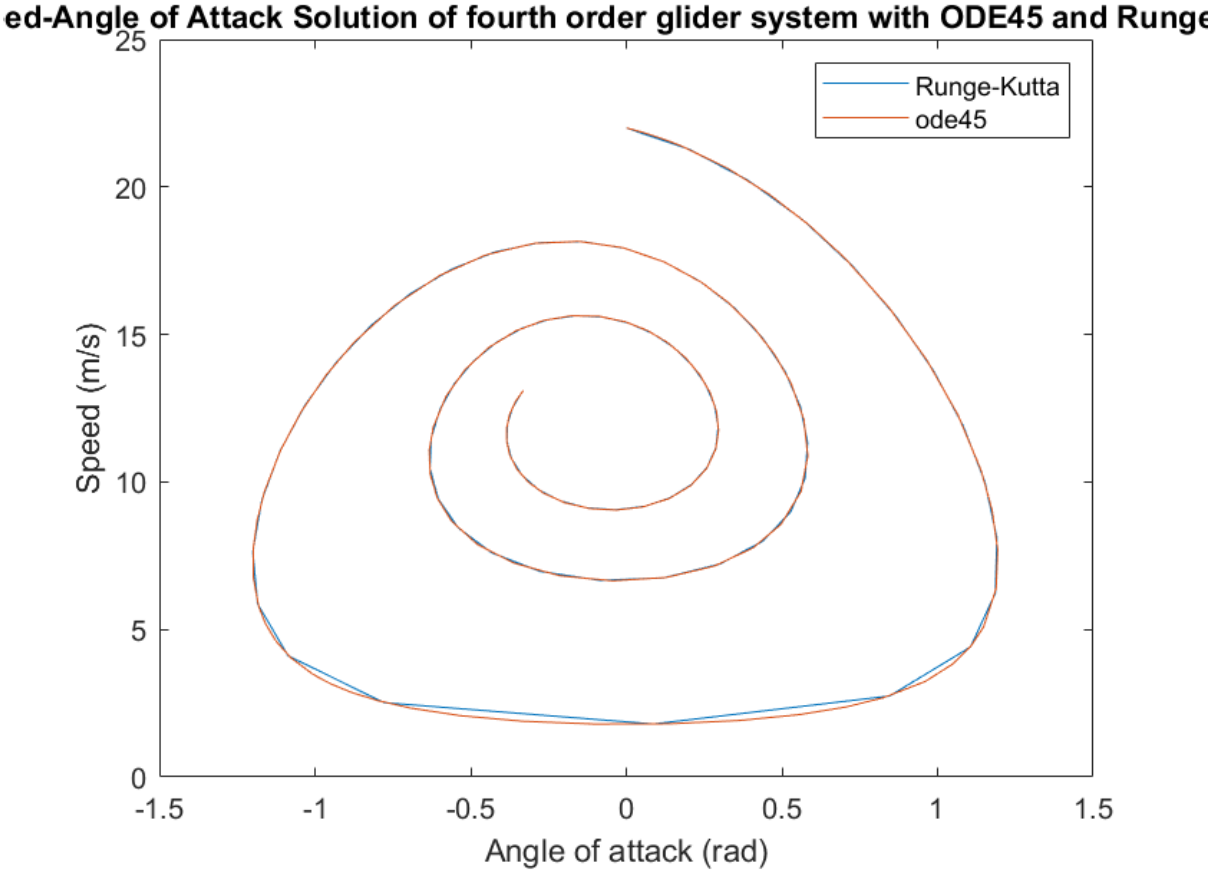
Below are the plots useful for comparing ode45 and Forward Euler methods results for the aforementioned initial condition:


Position Solution of fourth order glider system with ODE45 and FE
FE
ode45
Y location (m)
X location (m)
25
20
15
10
5
0
-5
0
20
40
60
80
100
120
140
160


Speed-Angle of Attack Solution of fourth order glider system with ODE45 and F
FE
ode45
Speed (m/s)
Angle of attack (rad)
25
20
15
10
5
0
-1.5
-1
-0.5
0
0.5
1
1.5

Further, plots for comparing ode45 (reference solution) with my implementation of RK4:





As expected RK4 is converging orders of magnitude faster than FE at a small enough timestep. I experimentally estimated the eigenvalues associated with this ODE system by computing the Jacobian matrix and determining the roots of $\det(\frac{\partial f}{\partial u} - \lambda I)$.

The determinant I computed for this problem for the expression in the parentheses, estimating the Jacobian given the f function (time derivative of state vector) from which we subtract the 4x4 identity matrix scaled by λ, is:

$$\det\left(\frac{\partial f}{\partial u} - \lambda I\right) = \begin{vmatrix} -2R_D v - \lambda & -g\cos(\theta) & 0 & 0 \\ R_L + \frac{g\cos(\theta)}{v^2} & \frac{g\sin(\theta)}{v} - \lambda & 0 & 0 \\ \cos(\theta) & -v\sin(\theta) & -\lambda & 0 \\ \sin(\theta) & v\cos(\theta) & 0 & -\lambda \end{vmatrix}$$

Obviously one could use Laplace expansion to find the explicit form of this determinant. After seeing that 0 is an eigenvalue of multiplicity two for this problem, one only has to determine the other two eigenvalues (which are complex conjugates). Dependences in v and AoA would disappear but computations are error prone. I wrote instead a Python script to represent the eigenvalues found in the complex plane:

```
import numpy as np
import matplotlib.pyplot as plt
```

```
#define constants
g=9.81
air_density=1.22
m=0.65
S=0.06
CD=0.10
CL=1.20
RD=air_density*CD*S/(2*m)
RL=air_density*CL*S/(2*m)
#define jacobian of the problem
def dfdu(v,theta,x=None,y=None):
    return np.array([[-2*RD*v,-
g*np.cos(theta),0,0],[RL+g*np.cos(theta)/(v**2),g*np.sin(theta)/v,0,0],[np.
cos(theta),-v*np.sin(theta),0,0],[np.sin(theta),v*np.cos(theta),0,0]])
def ut(v,theta):
    return np.array([[-g*np.sin(theta)-RD*v**2],[RL*v-
g*np.cos(theta)/v],[v*np.cos(theta)],[v*np.sin(theta)]])
#s=np.linalg.eig(dfdu(22,0))[0][3]
#print(s)
#print(np.real(s))
#print(np.imag(s))
imaginary_list=[0.0]
real_list=[0.0]
thlist=list(np.linspace(-np.pi/2,np.pi/2,10))
vlist=list(np.linspace(0.01,23,10))
for theta in thlist:
    for v in vlist:
        eigenvalues=list(np.linalg.eig(dfdu(22,0))[0])
        for eigenvalue in eigenvalues:
            if not (np.imag(eigenvalue)==0 and np.real(eigenvalue)==0):

imaginary_list.append(np.imag(eigenvalue));real_list.append(np.real(eigenva
lue))

plt.plot(real_list,imaginary_list, 'r.', label='Eigenvalues')
#plt.plot(al_7, al_load, 'b.', label='Aluminum Channel 5')
plt.title('Eigenvalue in Complex plane simulation graph')
plt.xlabel('Real axis')
plt.ylabel('Imaginary axis')
plt.legend()
plt.show()
print(imaginary_list)
print(real_list)
print()
print(dfdu(12,-0.08314))
print(ut(12,-0.08314))
```

Below you can see the locations of the eigenvalues in complex plane

The eigenvalues are:

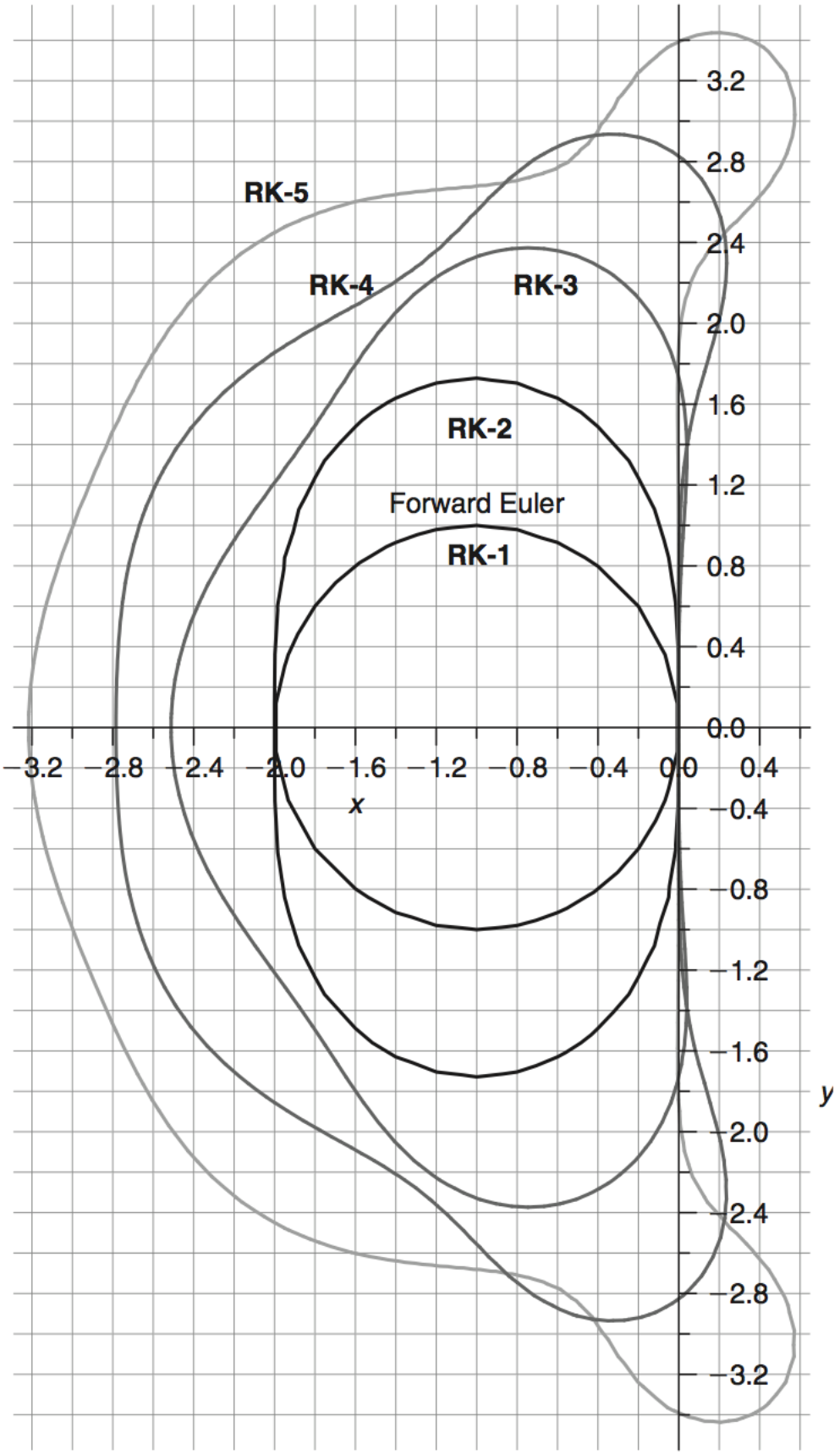
$$\lambda_1 = -0.12387692307692307 + j\ 0.9199693359729552$$

$$\lambda_2 = -0.12387692307692307 - j\ 0.9199693359729552$$

$$\lambda_3 = \lambda_4 = 0$$

As stated before, this problem will show decaying oscillations in almost all cases. It is a non-stiff problem because its spectral conditioning number is 1.

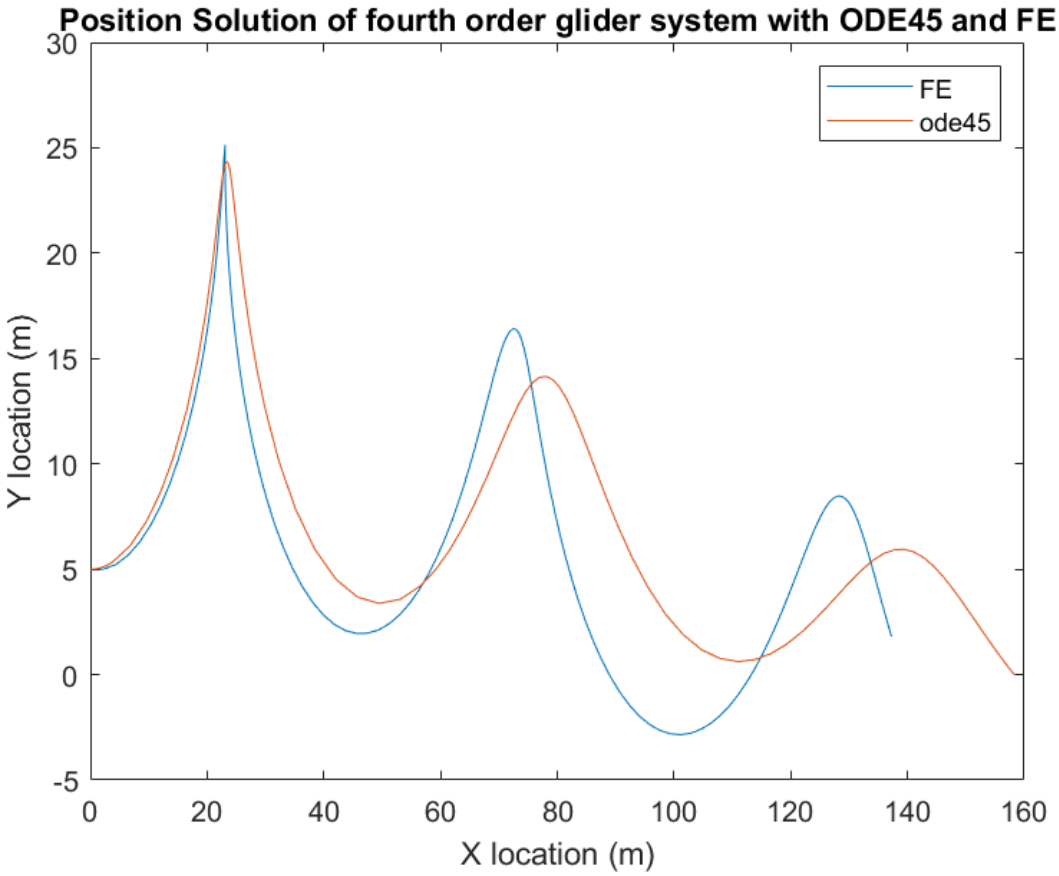
Now to determine some order of magnitude for timesteps that allow stability, use the following graph and consider the lines $\lambda_1 \Delta t$ and $\lambda_2 \Delta t$:

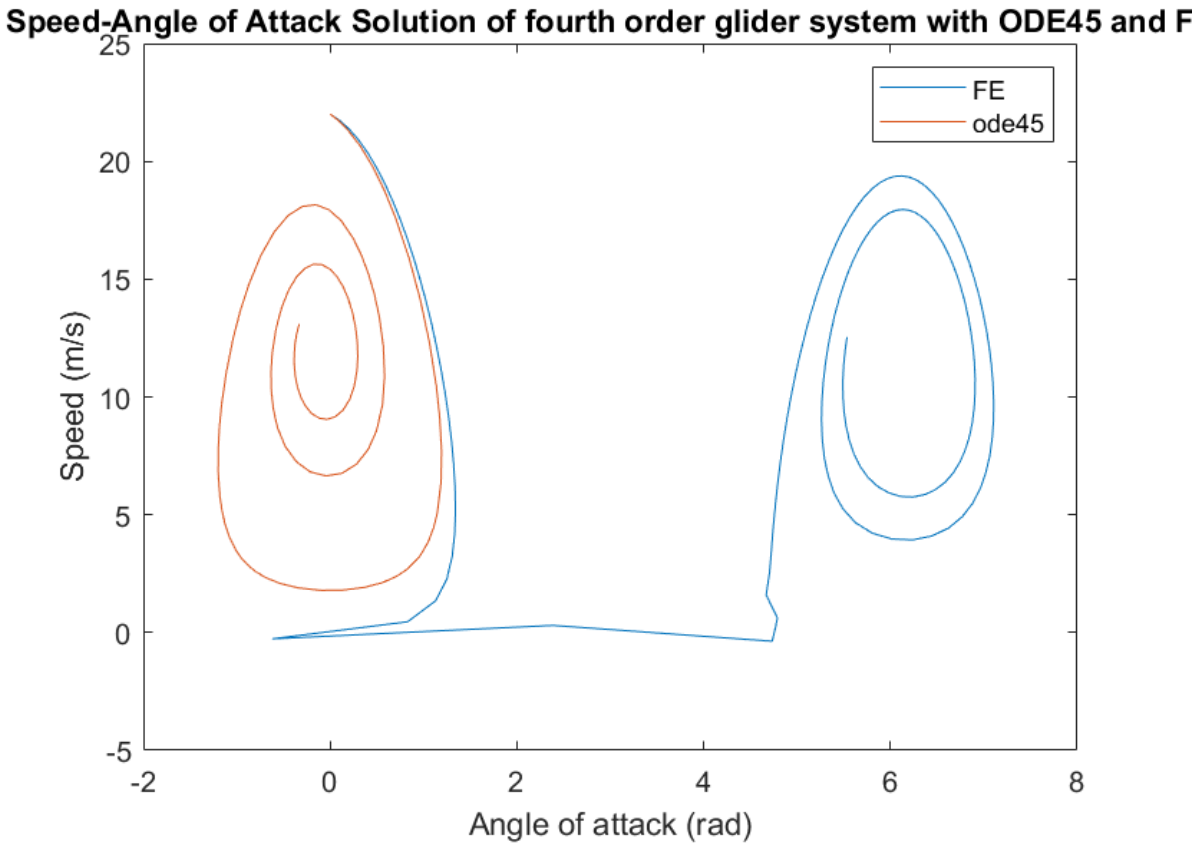

RK-5
RK-4
RK-3
RK-2
Forward Euler
RK-1
3.2
2.8
2.4
2.0
1.6
1.2
0.8
0.4
0.0
−0.4
−0.8
−1.2
−1.6
−2.0
−2.4
−2.8
−3.2
−3.2 −2.8 −2.4 −2.0 −1.6 −1.2 −0.8 −0.4 0.0 0.4
x
y

Graphically I estimated the maximum absolute value for the product $\lambda\Delta t$ (by following the line dictated by lambda in the complex plane) and then divided by the norm of the vector representing the eigenvalue in the complex plane. I estimated that the maximal timestep for FE is 0.2 seconds while for RK4 the maximum allowable time step should be around 2.0 seconds. Numerical simulations proved that I overestimated the maximum stable timestep for both methods. Apparently for this problem, FE is stable if the timestep is smaller than 0.11 seconds, while RK4 integration is stable for timesteps smaller than 1.8 seconds. Errors may be due to the method of measuring segments on the stability plot or because of the erroneous determination of eigenvalues for this ODE system.

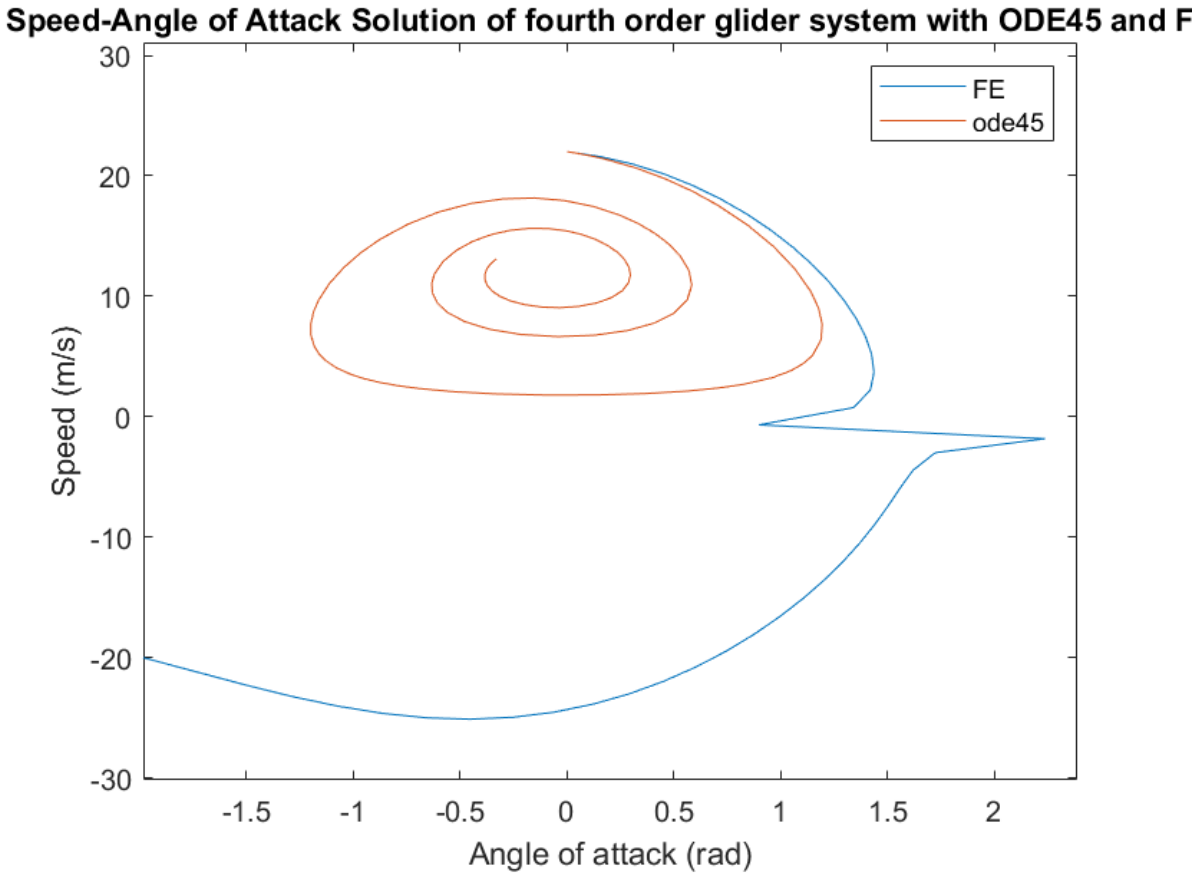
As you have seen in the plots between ode45 and RK4 there are almost no differences as long as the timestep is chosen in the interval that satisfies stability. For unstability situations I obtained the following plots:

FE with timestep of 0.1 seconds:

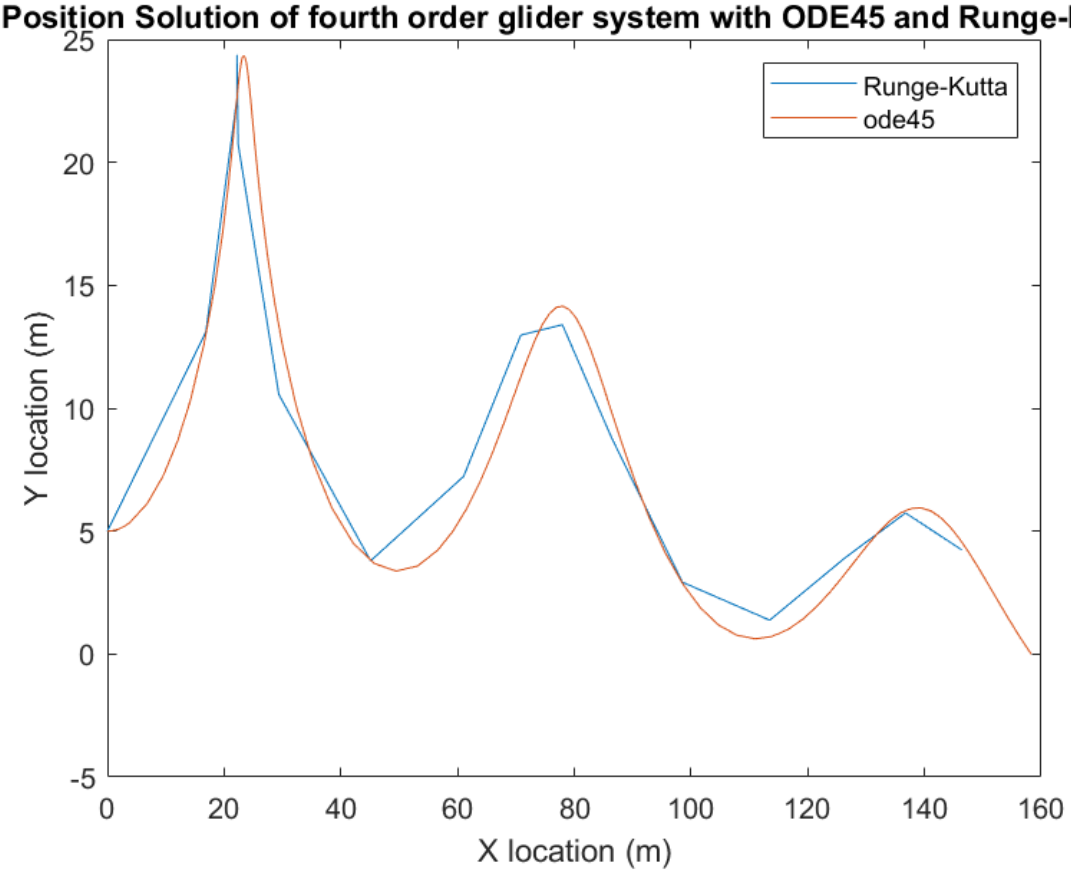


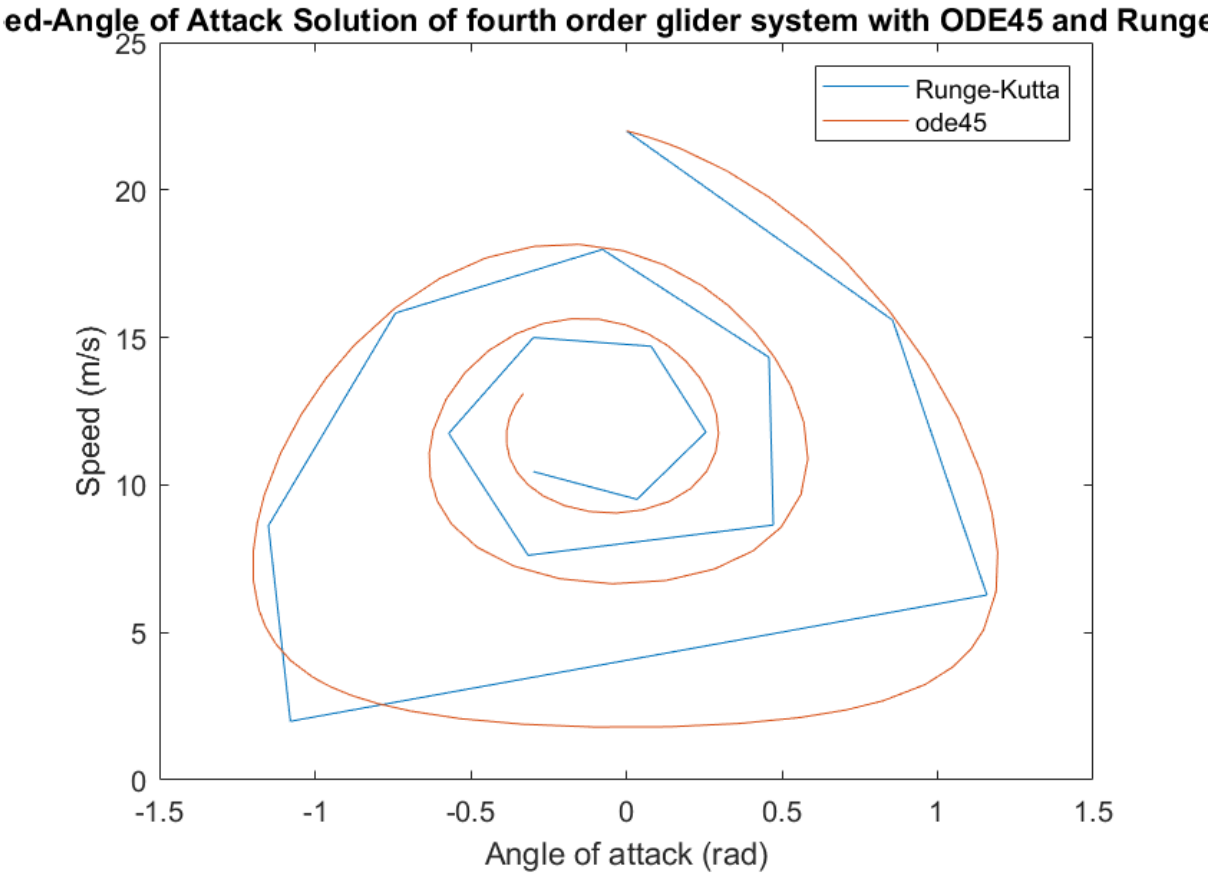


Below you can see FE iteration with timestep 0.15 seconds (for this one observe that the problem was caused by the increase of AoA over 90 degrees and starting a looping solution). There is actually just one loop in the solution, but all are shown because I have used the time limit for ode45 (that is FE was shown without a proper stopping condition):

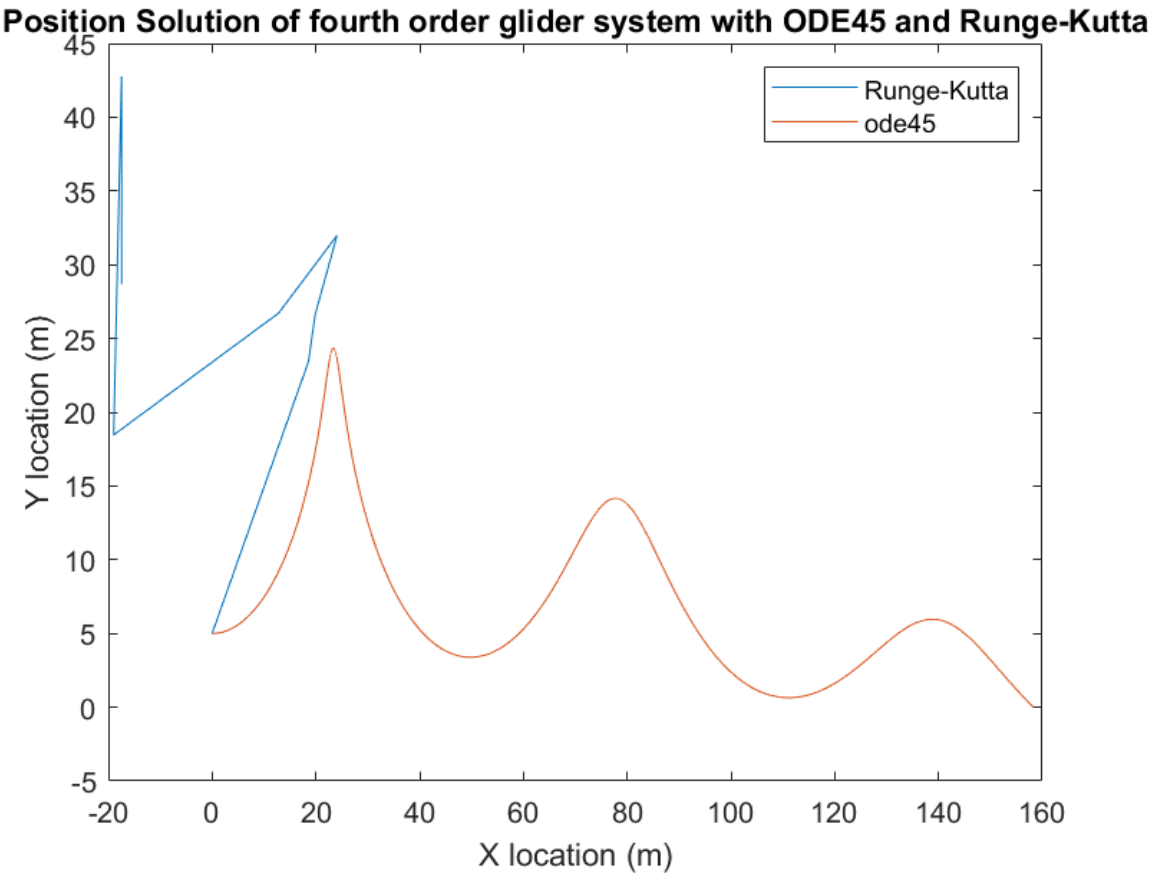


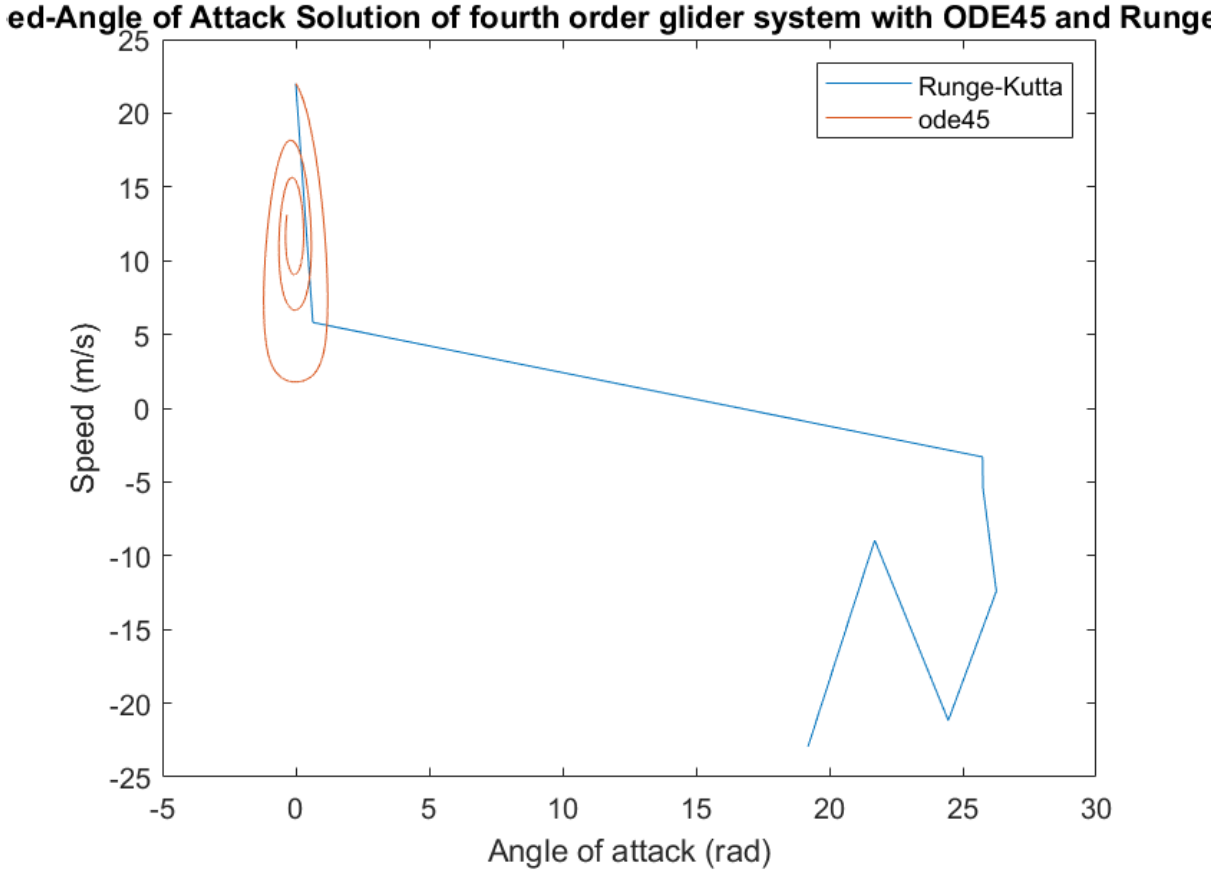
Here you can see RK4 solution with timestep of 1 second vs. ode45 (still stable, increasing the timestep more results in unstable solutions):


Position Solution of fourth order glider system with ODE45 and Runge-Kutta
Runge-Kutta
ode45
Y location (m)
X location (m)
25
20
15
10
5
0
-5
0
20
40
60
80
100
120
140
160

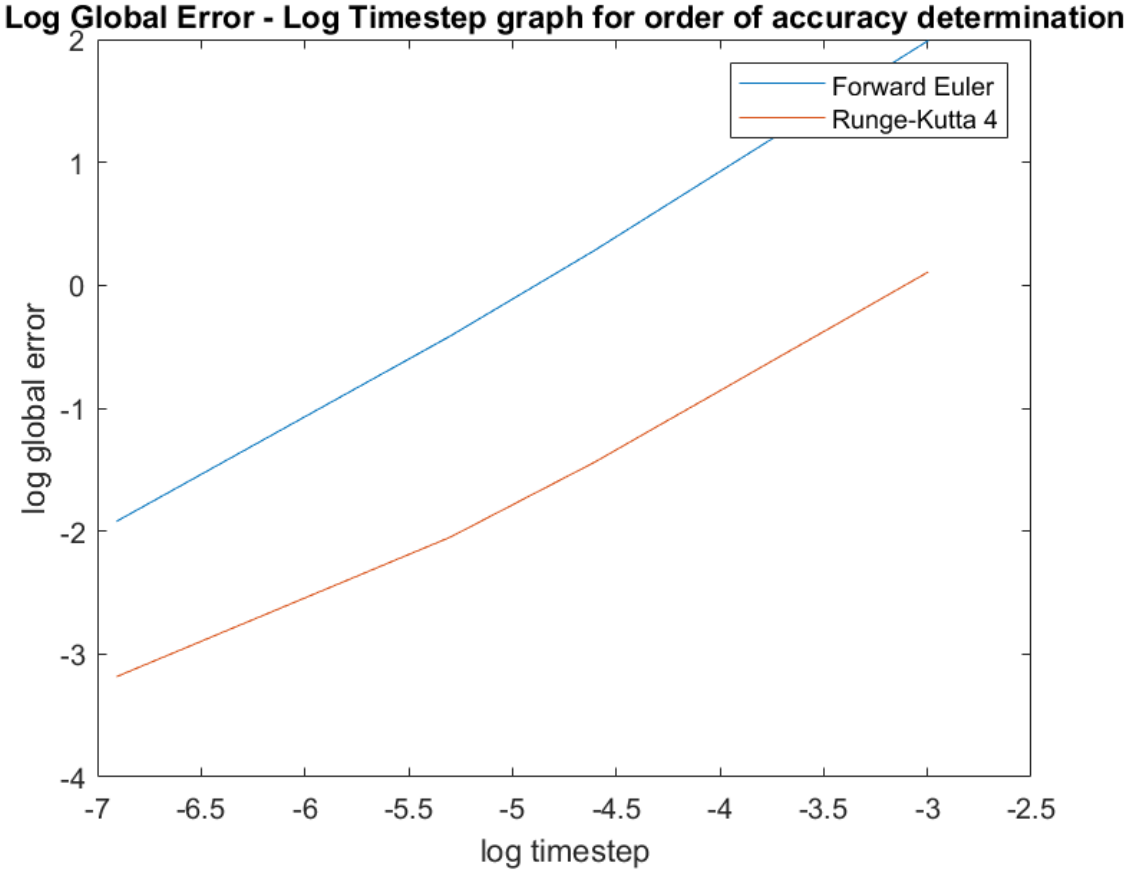

ed-Angle of Attack Solution of fourth order glider system with ODE45 and Runge
Runge-Kutta
ode45
Speed (m/s)
Angle of attack (rad)
25
20
15
10
5
0
-1.5
-1
-0.5
0
0.5
1
1.5

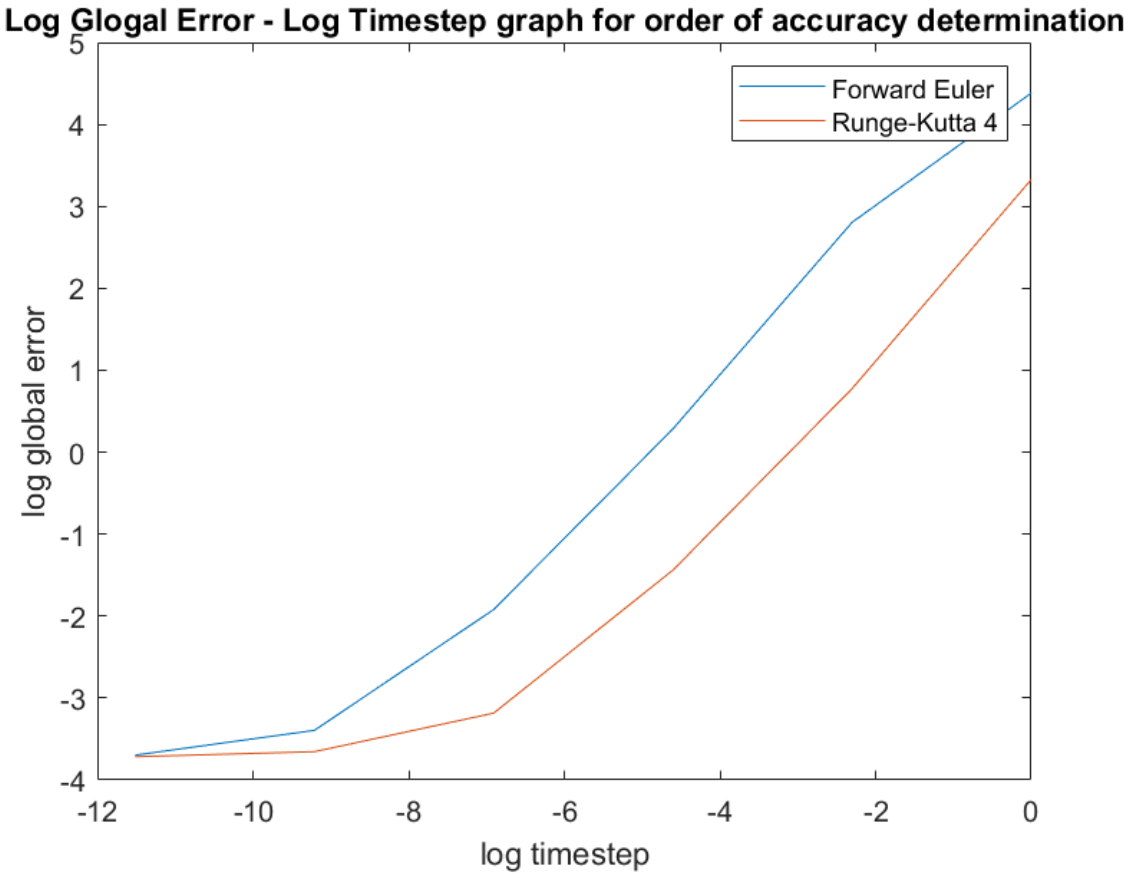
For a timestep of 2 seconds RK4 gives:




ed-Angle of Attack Solution of fourth order glider system with ODE45 and Runge
Runge-Kutta
ode45
Speed (m/s)
Angle of attack (rad)
25
20
15
10
5
0
-5
-10
-15
-20
-25
-5
0
5
10
15
20
25
30

3.


Log Global Error - Log Timestep graph for order of accuracy determination
Forward Euler
Runge-Kutta 4
log global error
log timestep
2
1
0
-1
-2
-3
-4
-7
-6.5
-6
-5.5
-5
-4.5
-4
-3.5
-3
-2.5

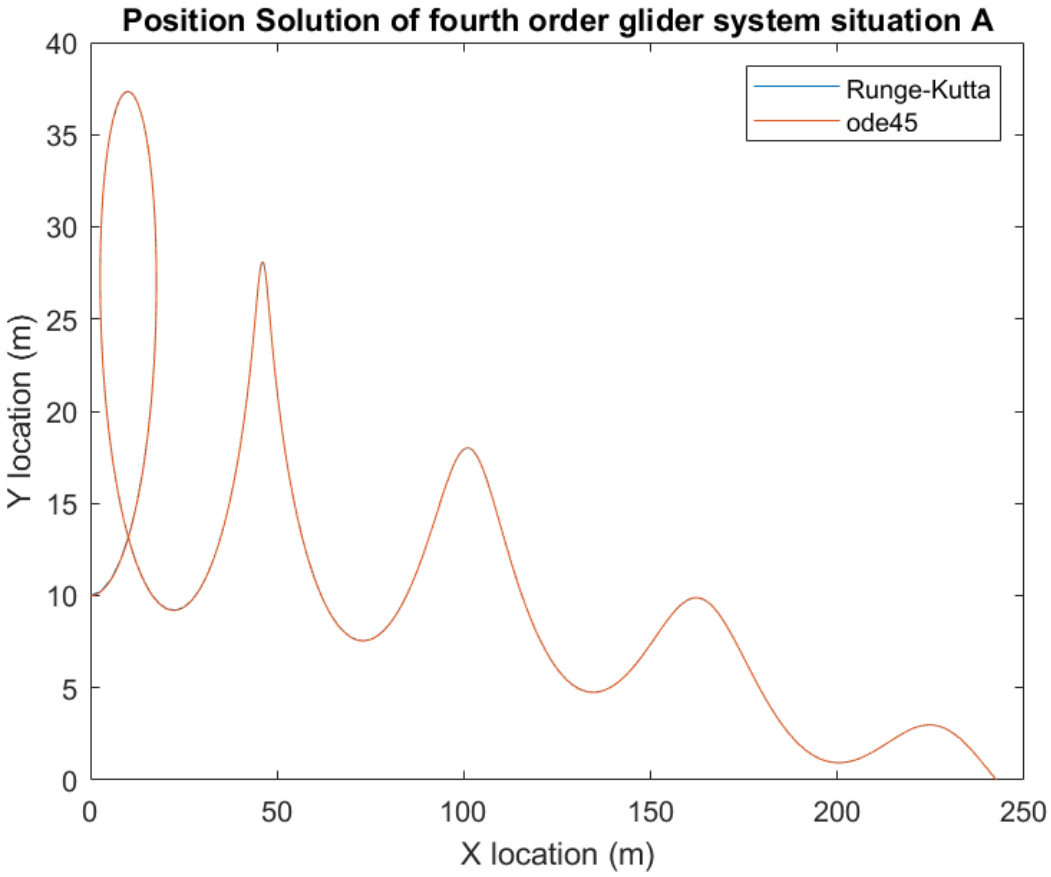

Log Glogal Error - Log Timestep graph for order of accuracy determination
Forward Euler
Runge-Kutta 4
log global error
log timestep
5
4
3
2
1
0
-1
-2
-3
-4
-12
-10
-8
-6
-4
-2
0

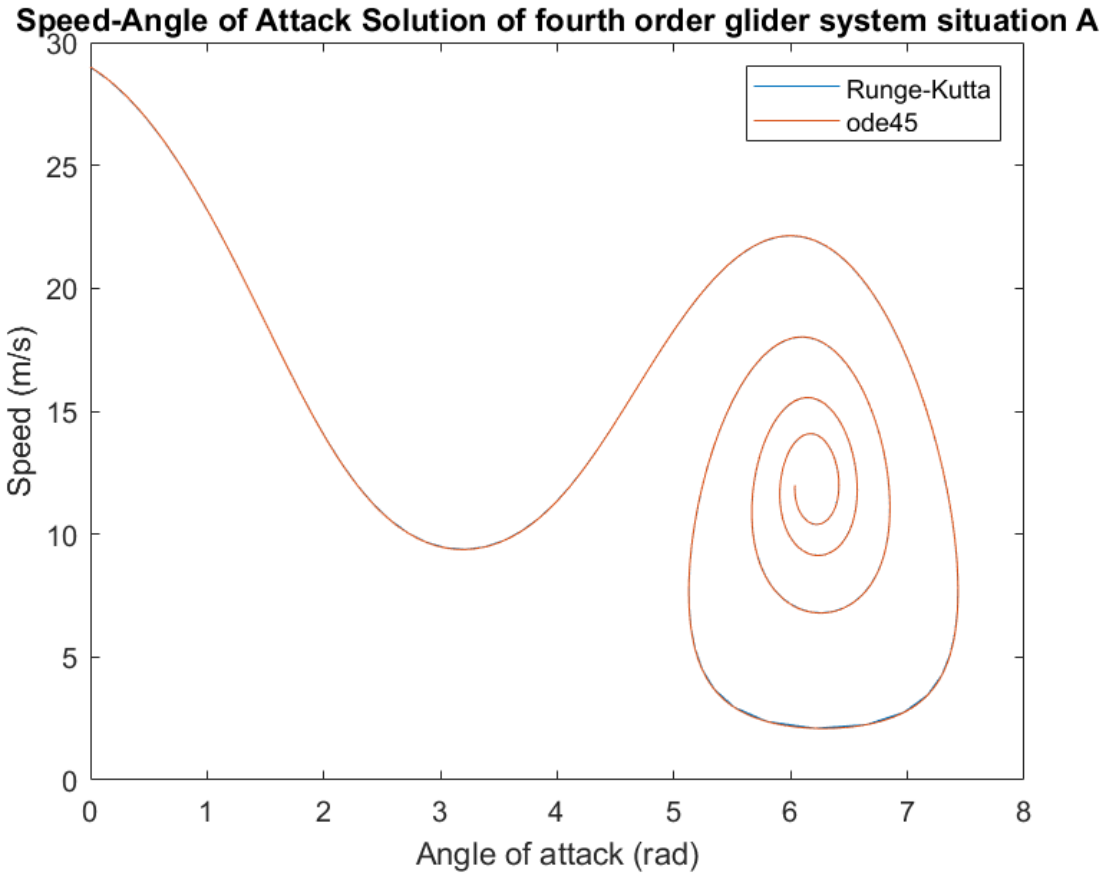
The results shown above are unexpected because Forward Euler method has the first order of accuracy while Runge-Kutta 4 has the 4-th order of accuracy. One would have expect the logarithmic plot to show the RK4 corresponding line as four times steeper than FE. Since the log of the relative error is somewhat large, I would suggest that the portion of my code comparing the found solutions to the "exact solution" is erroneous. The second line may be problematic if t_exact isn't calculated with a large enough resolution:

```
for i=2:length(t)
        [d,it]=min(abs(t_exact-i*dt)); %find element in t_exact
closest to i*dt
        err=norm(V(:,i)-u_exact(:,it));
        if err>max_err
         max_err=err;
```

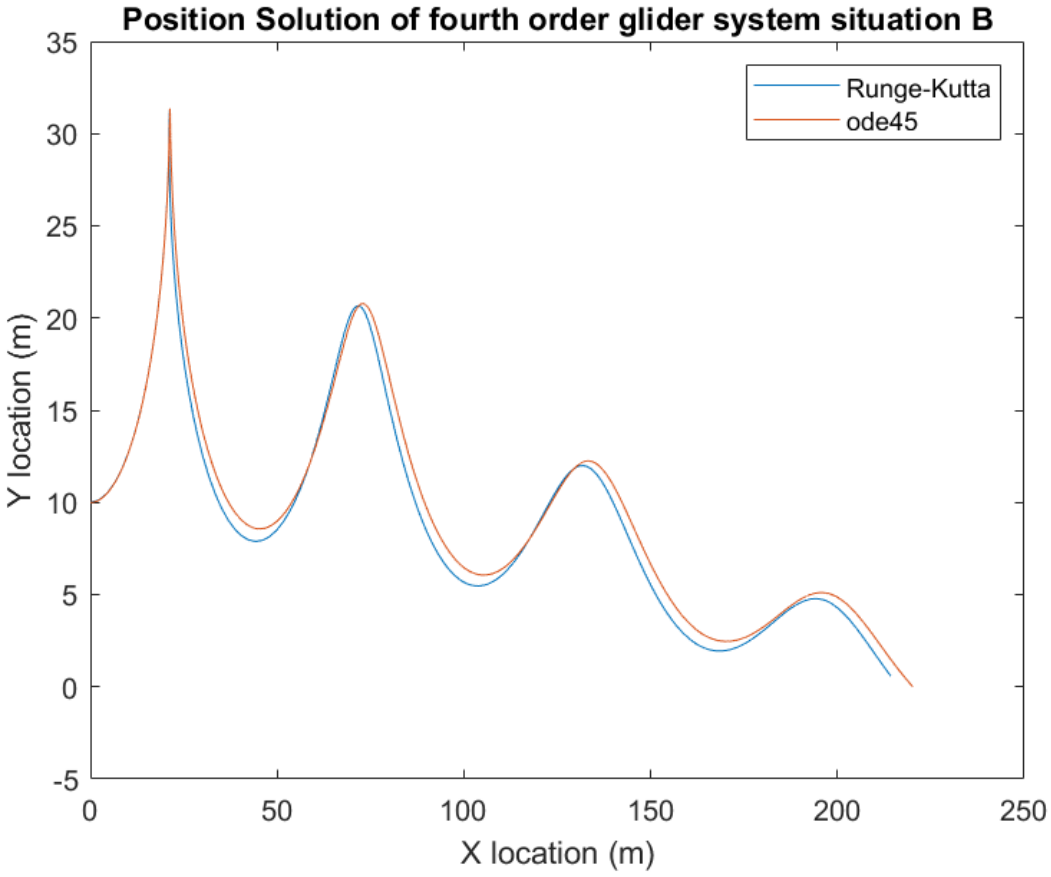
4. For all the plots below, the results obtained with my implementation of RK4 is shown along the ode45 reference with low tolerance.

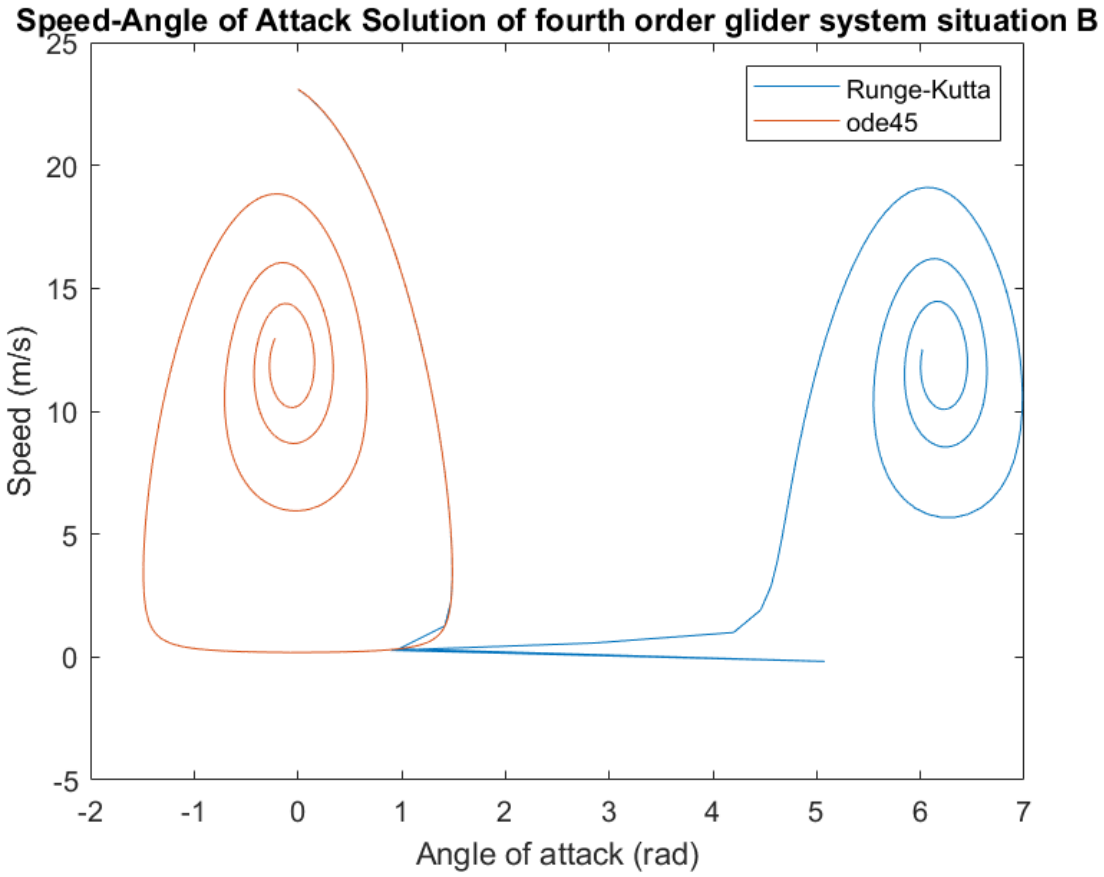
A. The initial speed is very high (29 m/s) so the initial lift is large, as the rate of increase of the angle of attack. The glider quickly gains altitude and at some point the angle of attack is nearly 90 degrees. In that moment the x-location of the glider starts decreasing and almost immediately the AOA increases to 180 degrees, and the glider continues its looping movement until it falls to 10 meter altitude with an angle of attack close to zero. Afterwards, simple oscillations are observed. In the AoA – speed space, we observe the initial large increase of the angle of attack from the distance between the spiral's attractor and the initial state in the space.


Position Solution of fourth order glider system situation A
Runge-Kutta
ode45
Y location (m)
X location (m)
0
5
10
15
20
25
30
35
40
0
50
100
150
200
250

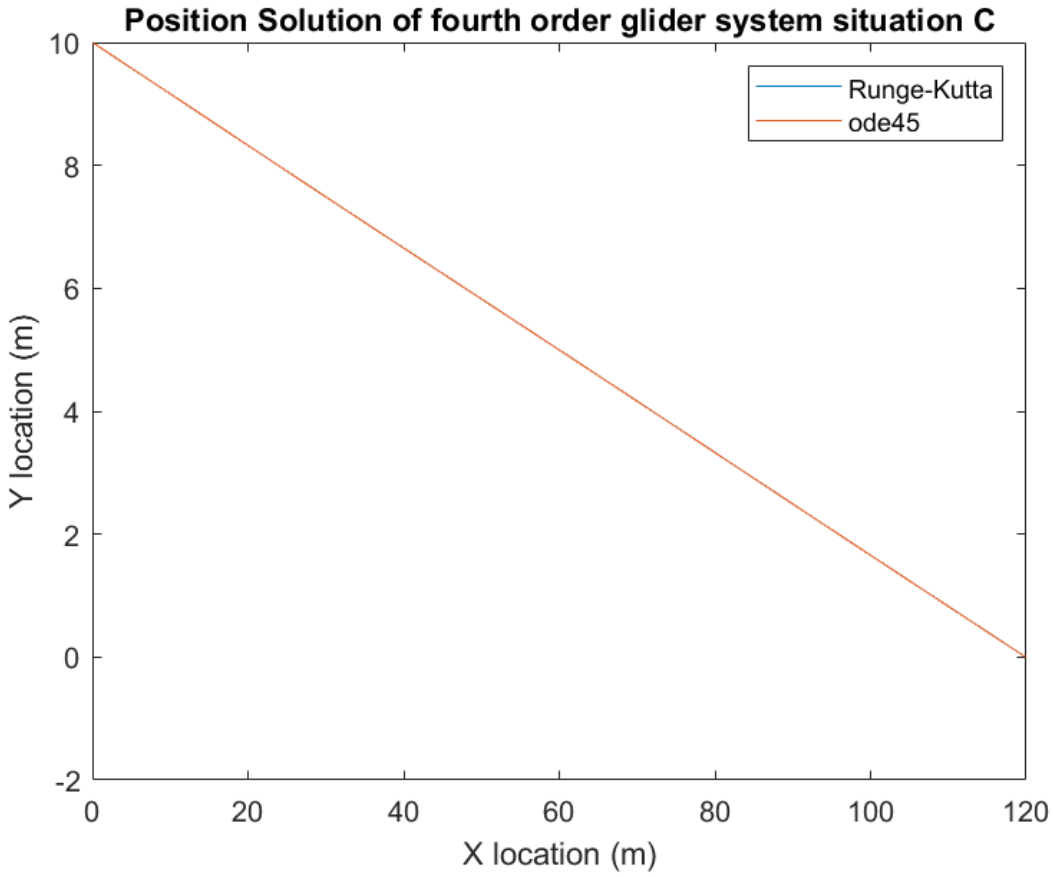

Speed-Angle of Attack Solution of fourth order glider system situation A
Runge-Kutta
ode45
Speed (m/s)
Angle of attack (rad)
0
5
10
15
20
25
30
0
1
2
3
4
5
6
7
8

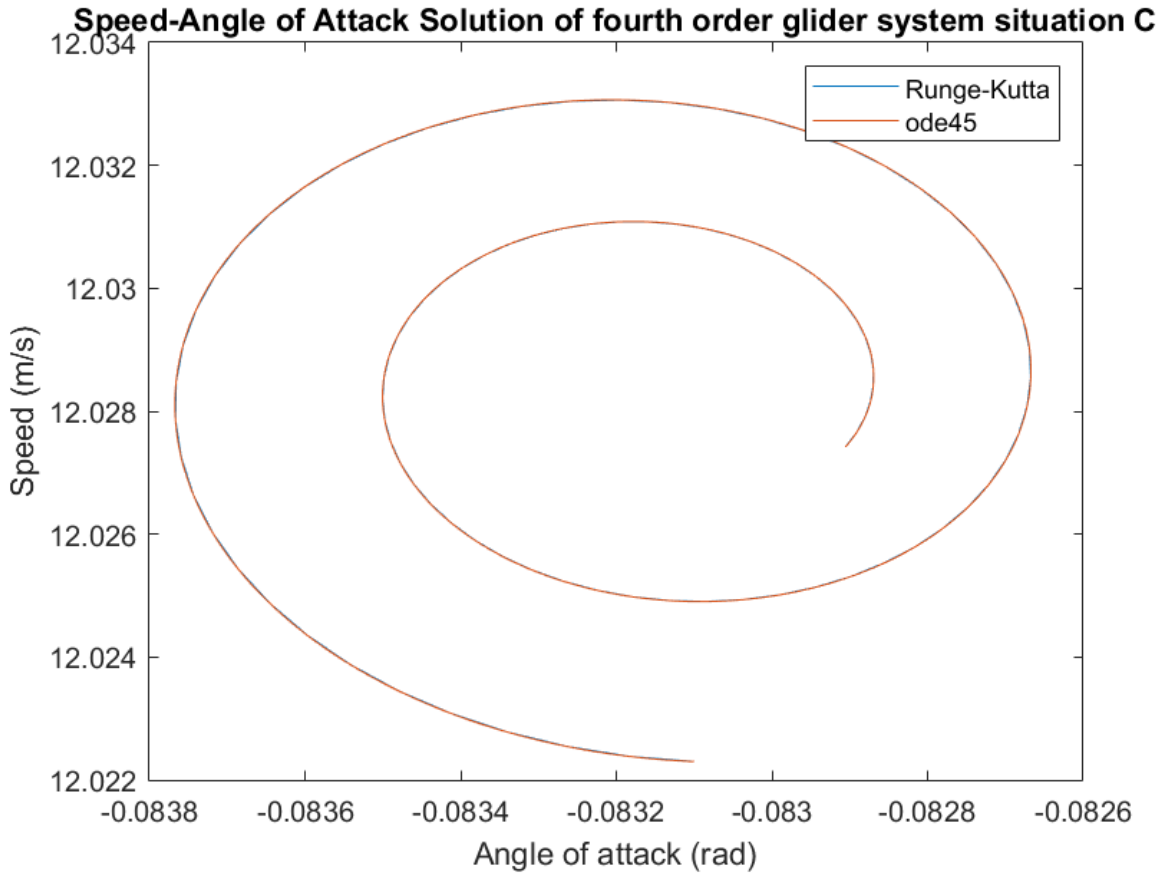
B. The main difference between this trajectory and the previous one is the slightly smaller initial speed. You can see an interesting behavior in the point where the rate of change of the angle of attack is maximal: because of the large timestep I used for RK4 integration, in the AoA space there is a major difference between the solution computed with ode45 and RK4. If we decrease the timestep of RK4 the difference observed is smaller. As you see the state corresponding to the highest altitude can be described as a source in the state-space, due to its intrinsic instability which may result in apparently chaotic behavior. If I would have used a larger timestep I would have seen a loop maneuver for this glider.



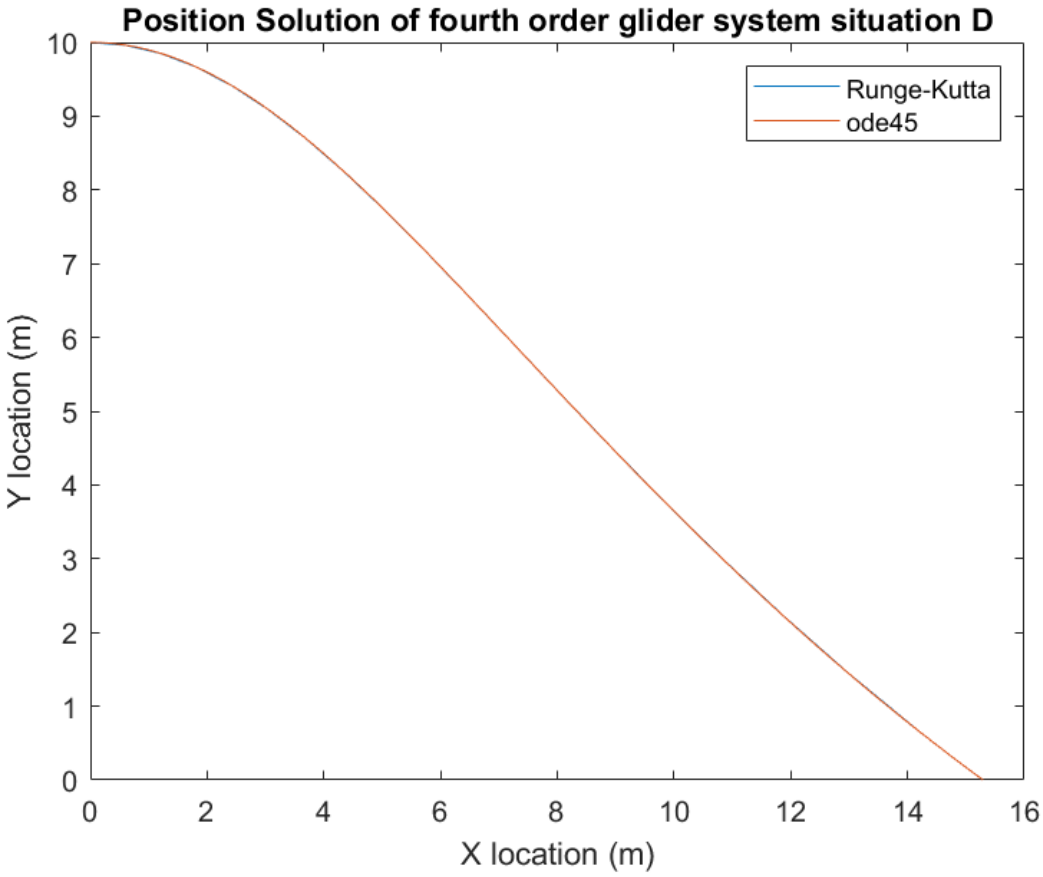


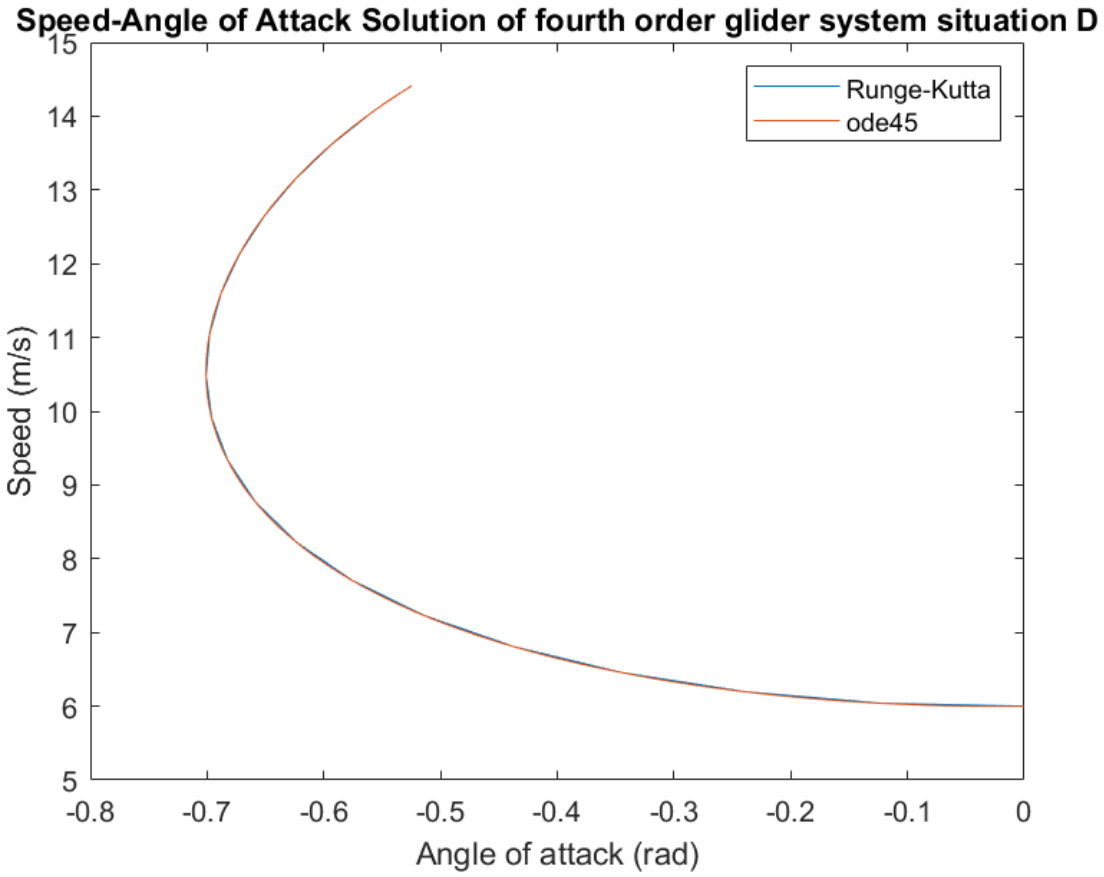
C. This initial state is has speed and angle of attack very close to what I have found at part 5 to be a sink. With this initialization, the glider is descending on a trajectory with constant angle of attack and speed. Compared to other cases, the glider is just losing altitude. You can see in the AoA-speed graph the sink behavior near the point of speed 12m/s and AoA=-0.08314 rad.


Position Solution of fourth order glider system situation C
Runge-Kutta
ode45
Y location (m)
X location (m)
10
8
6
4
2
0
-2
0
20
40
60
80
100
120


Speed-Angle of Attack Solution of fourth order glider system situation C
Runge-Kutta
ode45
Speed (m/s)
Angle of attack (rad)
12.034
12.032
12.03
12.028
12.026
12.024
12.022
-0.0838
-0.0836
-0.0834
-0.0832
-0.083
-0.0828
-0.0826

D. The initial speed is so small in this case that the angle of attack decreases quickly and the glider loses altitude. The movement is descending only. While falling, potential energy becomes (partially) kinetic energy so the lift increases, thus increasing the angle of attack. The model reaches the ground while the angle of attack is still negative. During this fall the travelled distance (on the x axis) is very small compared to the previous three cases.





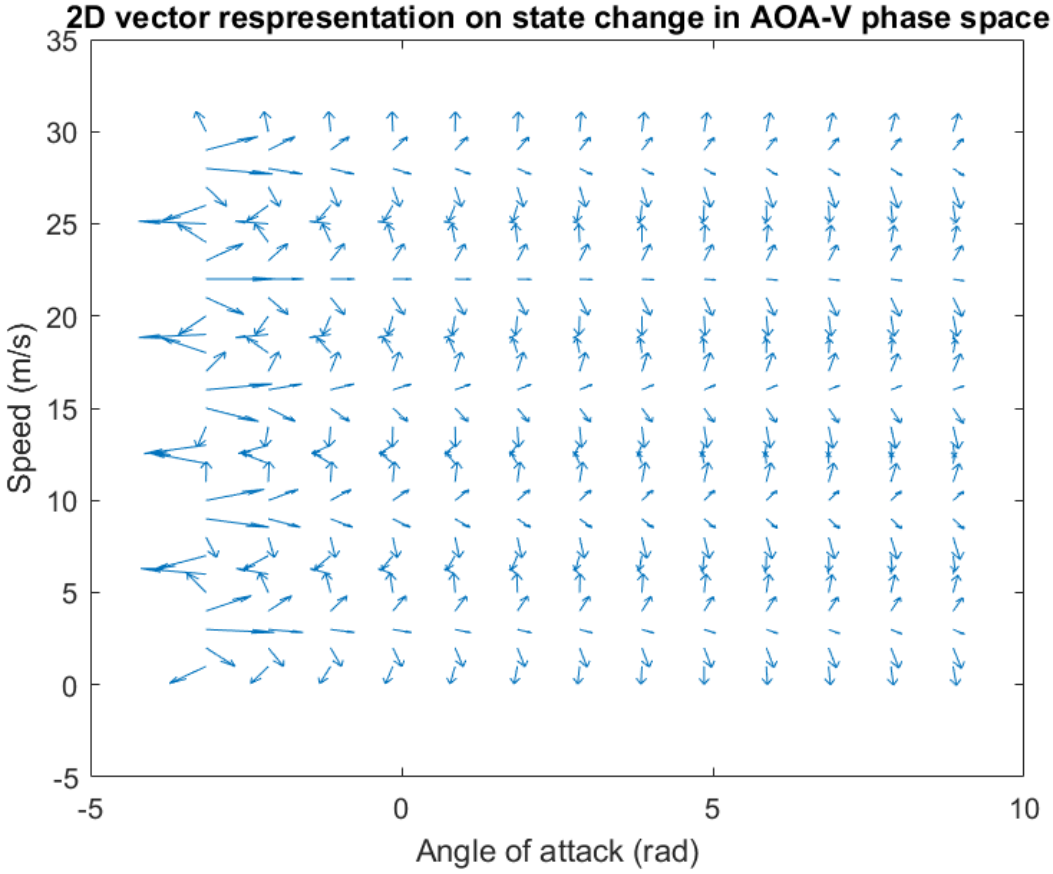
5. I have attached a hand-written Annex to this file containing the derivation of the expressions for (v^*, θ^*). I found that this point is a saddle and can be described by these mathematical expressions:

$$v^* = \sqrt{\frac{g}{\sqrt{R_D^2 + R_L^2}}}$$

$$\theta^* = arctg(-\frac{R_D}{R_L})$$

Thus we find that $(v^*, \theta^*) \approx (12\, ms^{-1}, -0.08314\, rad)$. The stability of this solution in the phase space on the first two coordinates suggest that in the long term the solution would have constant speed and angle of attack, and that the glider would evolve on a descending line (y decreases, x increases, and the rate of change of x-y coordinates is constant).

6.



The plot above suggests that when we start with state vector U_a from part 4, (29m/s and zero angle of attack) the speed decreases and the angle of attack increases.

5. $\frac{dV}{dt}(V^*,\theta^*) = -g\sin\theta^* - R_D V^{*2} = 0$

$\frac{d\theta}{dt}(V^*,\theta^*) = R_L V^* - \frac{g\cos\theta^*}{V^*} = 0 \rightarrow R_L V^{*2} = g\cos\theta^*$

$V^{*2} = \frac{g\cos\theta^*}{R_L}$

Prima ecuație devine $-g\sin\theta^* - \frac{R_D}{R_L} g\cos\theta^* = 0$; $-g\,tg\,\theta^* - \frac{R_D g}{L} = 0$

$g\,tg\,\theta^* + \frac{R_D g}{R_L} = 0 \rightarrow tg\,\theta^* = -\frac{R_D}{R_L}$; ~~$\theta^* = \pi$~~

$\boxed{\theta^* = \arctg\left(-\frac{R_D}{R_L}\right)}$ $V^* = \frac{g\cos\theta^*}{R_L}$

Știm $tg^2\theta^* = \frac{1-\cos^2\theta^*}{\cos^2\theta^*} \rightarrow tg^2\theta^*\cos^2\theta^* = 1-\cos^2\theta^*$, $1 = (tg^2\theta^*+1)\cos^2\theta^*$

$\cos\theta^* = \frac{1}{\sqrt{tg^2\theta^*+1}} \rightarrow V^{*2} = \frac{g}{R_L}\frac{1}{\sqrt{tg^2\theta^*+1}} = \frac{g}{\sqrt{R_D^2+R_L^2}}$

$\frac{1}{\sqrt{tg^2\theta^*+1}} = \frac{1}{\sqrt{\frac{R_D^2}{R_L^2}+1}} = \frac{R_L}{\sqrt{R_D^2+R_L^2}}$; $\boxed{V^{*2} = \frac{g}{\sqrt{R_D^2+R_L^2}}}$

Etimez $R_D = \frac{\rho C_D S}{2m} = C_D \cdot \frac{\rho S}{2m} = 0.00563$ $\frac{\rho S}{2m} = 0.056307$

$R_L = \frac{\rho C_L S}{2m} = C_L \cdot \frac{\rho S}{2m} = 0.06756923$

$\boxed{V^{*2} \approx 144\,m/s \,;\ \theta^* = -4.76^\circ}$ $(\theta^* = -0.08314\,rad)$

Analiză dim. $[R_D] = \frac{kg\,m^{-3}\,m^2}{kg} = m^{-1} \rightarrow [V]^* = \frac{m s^{-2}}{m^{-1}} = m^2 s^{-2}$

$V^* = \pm\sqrt{\frac{g}{\sqrt{R_D^2+R_L^2}}}$

$\boxed{V^* \approx \pm 12\,m s^{-1}}$

LS În cazul nostru $V^* > 0$ este relevant.

Describe local behavior with $(u^*+s) = \dot{u}^* + \left.\frac{\partial f}{\partial u}\right|_{u^*} s$. Use the Python source to estimate:

$\dot{u}^* = [0,\ 0,\ 1.196\times10^{1},\ -9.965\times10^{-1}]^T$

$\left.\frac{\partial f}{\partial u}\right|_{u^*} = \begin{bmatrix} -0.135 & -9.78 & 0.0 & 0.0 \\ 0.135 & -0.0679 & 0.0 & 0.0 \\ 0.997 & 0.9965 & 0.0 & 0.0 \\ -0.083 & 11.96 & 0.0 & 0.0 \end{bmatrix}$

(V^*,θ^*) is a ~~saddle~~ in the AoA-speed space